Network Working Group
Request for Comments: 973

Paul Mockapetris
ISI
January 1986

Domain System Changes and Observations

STATUS OF THIS MEMO

This RFC documents updates to Domain Name System specifications RFC-882 [1] and RFC-883 [2], suggests some operational guidelines, and discusses some experiences and problem areas in the present system. Distribution of this memo is unlimited.

This document includes all changes to the Domain System through January, 1986. Change notices and additional discussion are available online in file [USC-ISIB.ARPA]<DOMAIN>DOMAIN.CHANGES.

OVERVIEW

This memo is divided into four major sections:

"UPDATES" which discusses changes to the domain specification which are in widespread use and should be regarded as being part of the specification.

"OPERATION GUIDELINES" which suggests rules-of-thumb for using the domain system and configuring your database which are appropriate in most cases, but which may have rare exceptions.

"EXPERIENCES" which discusses some unusual situations and common bugs which are encountered in the present system, and should be helpful in problem determination and tuning.

"PROBLEM AREAS" which discusses some shortcomings in the present system which may be addressed in future versions.

UPDATES

This section discusses changes to the specification which are final, and should be incorporated in all domain system software.

TTL timeouts too small

The 16 bit TTL field in RRs could not represent a large enough time interval. The 16 bit field, using seconds for units, has a maximum period of approximately 18 hours.

All time values, including all TTLs and the MINIMUM field of the SOA RR, are expanded to 32 bits.

CLASS changes

Class 2, originally reserved for CSNET, is obsolete. Class 3 has been assigned for use by CHAOS.

CNAME usage

The specification allows CNAME RRs to exist with other RRs at the same node. This creates difficulties since the other RRs stored with the CNAME at the alias might not agree with the RRs stored at the primary name.

If a node has a CNAME RR, it should have no other RRs.

* semantics

The use of * to represent a single label wildcard, along with the possibility of multiple * labels, led to difficult server implementations and complicated search algorithms. There were also questions regarding whether a * based specification could refer to names that were not contained in the zone which had the * specification.

While we might want the "inheritability" for some cases, it leads to implementation difficulties. The first of these is that whenever we can't find a RR in a particular zone, we have to search all parent zones to look for a suitable * result. (Alternatively we could develop some automatic method for insuring consistency or insist on careful duplication of inherited data.) We also must deal with conflicts, i.e. what if a subdomain doesn't want to inherit defaults.

Given these difficulties, the solution is to insist that delegation of authority cancels the * defaults. This is quite simple to implement; all you need to do is to check for delegation before looking for * RRs.

A second difficulty is the restriction that * match a single label. Thus if a name server is looking for RRs for the name A.B.C.D.E.F, it must check for *.B.C.D.E.F, *.*.C.D.E.F, *.*.*.D.E.F, etc. This check must also be careful of zone boundaries and multiplies the effort to handle a query.

The solution adopted is to allow a single * label in the leftmost part of a name stored in a zone, and to allow this label to match

any number of unknown labels or a single known label in the query name. However, the * match is only taken for parts of the tree which are neither delegated or explicitly represented.

The algorithm for performing the search in a tree structured database has the following steps:

1) Descend in the tree matching labels from right to left. If a delegation is found return that; if the specified node is found go to step 2, if the tree ends go to step 3.

2) Look for RRs that answer the query. If any are found, return them as the answer. If none are found, look for answers in a * node which has the same name as the query name except for the rightmost label. (e.g. if you can't find an answer at F.ISI.ARPA, look for a RR at *.ISI.ARPA)

3) The search for a desired name has failed; look for a node whose name is * plus however much matched. Look for answers there. (e.g. If you are looking for X.Y.ISI.ARPA and the tree ends at ISI.ARPA, look at *.ISI.ARPA. The same thing holds for Y.ISI.ARPA, or any name of the form <anything>.Z.ISI.ARPA, where Z is a label that doesn't exist under ISI.ARPA)

Note that this interpretation means that * matches names that are not in the tree, no matter how much of the tree is missing, and also matches one level's worth of known tree.

AA semantics

When a name server is responding to a query for a particular name and finds a CNAME, it may optionally restart the search at the canonical name. If the server uses the restart feature, the answer section of the returned query contains one (or more) CNAMEs, possibly followed by answers for the primary name. The canonical name will usually be in the same zone as the alias, but this need not be the case. If the server is authoritative for one of the names but not both, it is not clear whether the AA bit should be set.

The solution adopted is to make the AA refer to the original query name.

Master file format

The present specification uses a somewhat awkward method for representing domain names in master files.

The change adopted is that all domain names in this file will be represented as either absolute or relative. An absolute domain name ends with a ".". A free standing "." is assumed to refer to the root. A relative domain name doesn't end with a dot, and is assumed to be relative to the current origin.

SERIAL number size

If the master file changes rapidly, an infrequently updated copy may miss the wrapping of the sequence number in the SERIAL field of the SOA, or misinterpret the number of updates that have taken place.

The SERIAL field is increased to 32 bits.

MD and MF replaced by MX

The original specification uses MD and MF RRs for mail agent binding. The problem is that a mailer making a MAILA query, which asks for both types, can't use the cache since the cache might have the results for a MD or MF query. That is, the presence of one of these types of information in the cache doesn't imply anything about the other type. The result was that either mailers would have to always consult authoritative servers or try to use partial information; neither of these is really acceptable.

The change is to replace MD and MF with a new type of RR called MX which conveys similar information in a single RR type. MX has been assigned a type code of 15 decimal. The format of the MX RR is a 16 bit preference value followed by a domain name. A node may have multiple MX RRs, and multiple MX RRs with the same preference value are allowed at a given node.

The preference values denote the relative preference that the mail destination places on the mail agents, with lower values being "better". A mailer is expected to at least try the mail agent(s) with the lowest preference value. The significance of particular preference values, the units of preference, and the linearity of preference values are not defined but left open; preference values should only be used to establish relative rankings.

For example, the current RRs:

```
MAIL-ORG    MD    HOST1
            MD    HOST2
            MF    HOST3
```

might be replaced by:

```
MAIL-ORG    MX    10 HOST1
            MX    10 HOST2
            MX    20 HOST3
```

The values 10 and 20 have no significance other than 10<20. A detailed discussion of the use of MX is the subject of [3].

Zone transfer

The original specification states that zone transfers take place in breadth first order. The intent was to make the transfer easier for the accepting name server to handle. This now doesn't work out to be very helpful, and is a severe pain for implementers using various hashing algorithms. The new rule is that you can transmit the records in any order you choose, so long as the SOA node of the zone is transmitted first and last, and no other duplication occurs.

IN-ADDR domain renamed

The name of the IN-ADDR domain is now IN-ADDR.ARPA. This change was made because many felt that the use of a top-level name was inappropriate to network-specific information.

OPERATIONAL GUIDELINES

This section suggests rules-of-thumb for using the domain system and configuring your database which are appropriate in most cases, but which may have rare exceptions.

Zone delegation

When a domain wishes to become independent from its parent, the RRs which mark the delegation in the parent and child zones should be carefully synchronized to minimize the possibility that resolvers become confused.

For example, suppose that we wish to create a new zone called ISI.EDU under an existing EDU zone, and that the servers for the child zone are X.ISI.EDU and Y.GOV.

We might add the following to the parent zone:

```
ISI.EDU.        10000 NS  X.ISI.EDU.
                10000 NS  Y.GOV.
X.ISI.EDU.      10000 A   <address of X.ISI.EDU.>
Y.GOV.          10000 A   <address of Y.GOV.>
```

and the following to the child zone:

```
ISI.EDU.        10000 NS  X.ISI.EDU.
                10000 NS  Y.GOV.
                50000 SOA <SOA information>
X.ISI.EDU.      10000 A   <address of X.ISI.EDU.>
Y.GOV.          10000 A   <address of Y.GOV.>
```

Note the following:

In both cases, the A RR for Y.GOV is included, even though Y.GOV isn't in the EDU or ISI.EDU domains. This RR isn't authoritative, but is included to guarantee that the address of Y.GOV is passed with delegations to it. Strictly speaking this RR need not be in either zone, but its presence is recommended. The X.ISI.EDU A RR is absolutely essential. The only time that a server should use the glue RRs is when it is returning the NS RRs and doesn't otherwise have the address of the server. For example, if the parent server also was authoritative for GOV, the glue RR would typically not be consulted. However, it is still a good idea for it to be present, so that the zone is self-sufficient.

The child zone and the parent zone have identical NS RRs for the ISI.EDU domain. This guarantees that no matter which server is asked about the ISI.EDU domain, the same set of name servers is returned.

The child zone and the parent zone have A RRs for the name servers in the NS RRs that delegate the ISI.EDU domain. This guarantees that in addition to knowing the name servers for the ISI.EDU domain, the addresses of the servers are known as well.

The TTLs for the NS RRs that delegate the ISI.EDU domain and the A RRs that represent the addresses of the name servers are all the same. This guarantees that all of these RRs will timeout simultaneously. In this example, the value 10000 has no special significance, but the coincidence of the TTLs is significant.

These guidelines haven't changed any of the flexibility of the system; the name of a name server and the domains it serves are still independent.

It might also be the case that the organization called ISI wanted to take over management of the IN-ADDR domain for an internal network, say 128.99.0.0. In this case, we would have additions to the parent zone, say IN-ADDR.ARPA.

We might add the following to the parent zone:

```
99.128.IN-ADDR.ARPA.  2000 NS   Q.ISI.EDU.
                      2000 NS   XX.MIT.EDU.
Q.ISI.EDU.            2000 A    <address of Q.ISI.EDU.>
XX.MIT.EDU.           2000 A    <address of XX.MIT.EDU.>
```

and the following to the child zone:

```
99.128.IN-ADDR.ARPA.  2000 NS   Q.ISI.EDU.
                      2000 NS   XX.MIT.EDU.
                      5000 SOA  <SOA information>
Q.ISI.EDU.            2000 A    <address of Q.ISI.EDU.>
XX.MIT.EDU.           2000 A    <address of XX.MIT.EDU.>
```

SOA serials

The serial field of the SOA RR for a domain is supposed to be a continuously increasing (mod 2**32) value which denotes the

version of the database. The idea is that you can tell that a zone has changed by comparing serial numbers. When you change a zone, you should increment the serial field of the SOA.

All RRs with the same name, class, and type should have the same TTL.

The logic here is that all of them will timeout simultaneously if cached and hence the cache can be reliably used.

Case consistency

The domain system is supposed to preserve case, but be case insensitive. However, it does nobody any good to put both RRs for domain name xxx and XXX in the data base - It merely makes caching ambiguous and decreases the efficiency of compression. This consistency should also exist in the duplicate RRs that mark delegation in the delegator and delegatee. For example, if you ask the NIC to delegate UZOO.EDU to you, your database shouldn't say uzoo.edu.

Inappropriate use of aliases

Canonical names are preferred to aliases in all RRs. One reason is that the canonical names are closer to the information associated with a name. A second is that canonical names are unique, and aliases are not, and hence comparisons will work.

In particular, the use of aliases in PTR RRs of the IN-ADDR domain or in NS RRs that mark delegation is discouraged.

EXPERIENCES

This section discusses some unusual situations and common bugs which are encountered in the present system, and should be helpful in problem determination and tuning. Put differently, you should try to make your code defend against these attacks, and you should expect to be the object of complaint if you make these attacks.

UDP addresses

When you send a query to a host with multiple addresses, you might expect the response to be from the address to which you sent the query. This isn't the case with almost all UNIX implementations.

UDP checksums

Many versions of UNIX generate incorrect UDP checksums, and most ignore the checksum of incoming UDP datagrams. The typical symptom is that your UNIX domain code works fine with other UNIXes, but won't communicate with TOPS-20 or other systems. (JEEVES, the TOPS-20 server used for 3 of the 4 root servers, ignores datagrams with bad UDP checksums.)

Making up data

There are lots of name servers which return RRs for the root servers with 99999999 or similar large values in the TTL. For example, some return RRs that suggest that ISIF is a root server. (It was months ago, but is no longer.)

One of the main ideas of the domain system is that everybody can get a chunk of the name space to manage as they choose. However, you aren't supposed to lie about other parts of the name space. Its OK to remember about other parts of the name space for caching or other purposes, but you are supposed to follow the TTL rules.

Now it may be that you put such records in your server or whatever to ensure a server of last resort. That's fine. But if you export these in answers to queries, you should be shot. These entries get put in caches and never die.

Suggested domain meta-rule:

> If you must lie, lie only to yourself.

PROBLEM AREAS

This section discusses some shortcomings in the present system which may be addressed in future versions.

Compression and types

The present specification attempts to allow name servers and resolvers to cache RRs for classes they don't "understand" as well as to allow compression of domain names to minimize the size of UDP datagrams. These two goals conflict in the present scheme since the only way to expand a compressed name is to know that a name is expected in that position.

One technique for addressing this problem would be to preface binary data (i.e. anything but a domain name) with a length octet.

The high order two bits of the length octet could contain either 01 or 10, which are illegal for domain names. To compensate for the additional bytes of data, we could omit the RDATA length field and terminate each RR with a binary length field of zero.

Caching non-existent names

In the present system, a resolver has no standard method for caching the result that a name does not exist, which seems to make up a larger than expected percentage of queries. Some resolvers create "does not exist" RRs with TTLs to guarantee against repetitive queries for a non-existent name.

A standard technique might be to return the SOA RR for the zone (note that only authoritative servers can say name does not exist) in the reply, and define the semantics to be that the requester is free to assume that the name does not exist for a period equal to the MINIMUM field of the SOA.

Cache conflicts

When a resolver is processing a reply, it may well decide to cache all RRs found in sections of the reply. The problem is that the resolver's cache may already contain a subset of these RRs, probably with different TTLs.

If the RRs are from authoritative data in the answer section, then the cache RRs should be replaced. In other cases, the correct strategy isn't completely clear. Note that if the authoritative data's TTL has changed, then the resolver doesn't have enough information to make the correct decision in all cases.

This issue is tricky, and deserves thought.

REFERENCES

[1] Mockapetris, P., "Domain Names - Concepts and Facilities", RFC-882, USC Information Sciences Institute, November 1983.

[2] Mockapetris, P., "Domain Names - Implementation and Specification", RFC-883, USC Information Sciences Institute, November 1983.

[3] Partridge, C., "Mail Routing and the Domain System", RFC-974, CSNET-CIC, BBN Laboratories, January 1986.